ADORDD
****

ADORDD for (x)Harbour its ready!

PLEASE MAKE A DONATION!

We will all profit for sure from such development so I think it's fair to ask everyone to contribute with a minimal importance of 20 Euros (because of the PayPal costs) for a good cause.

Go to http://ajusera.com/ scroll down and click PayPal button "Doar"

Or

You have bank details TRANSFERÊNCIA BANCÁRIA = BANK TRANSFER

You can transfer to those account details

They can send you a contribution receipt if you send them a email with your details.

This is a non-profit organization that performs an important social work supporting the elderly, young problematic, food distribution to the needy, etc. This organization is based on the work of Father Jeronimo Usera (Spanish) well known in Spain, Portugal and South America.

You can get more information at ajusera.com (sorry its only in portuguese)

Steps to start working with **ADORDD**:

==========

Step 1: Just add adordd.prg to your project and include adordd.ch

Step 2: Set adordd parameters (see in adordd.ch for syntax) and add it in your main.prg like this:

```
// Load ADORDD
REQUEST ADORDD, ADOVERSION, RECSIZE
RddRegister( "ADORDD",1 )
RddSetDefault( "ADORDD" )
```

A) SET ADODBF TABLES INDEX LIST TO…

It must contain all your actual indexes and expressions per table.

Array Spec :

[{ TableName, {Tag Name 1, Index expression, For Condition, lUnique, lDescend }, { Tag name 2 …..} }, { Table name 2 …. },{ Tag Name 1…… } } }

Example:

```
SET ADODBF TABLES INDEX LIST TO {
{"TABLE1",{"FIRST","FIRST"} }, {"TABLE2"
,{"CODID","STR(CODID,2,0)"}} }
```

In case table or tagname its build dynamically for ex tablename+Userid() the table must be place as tablename and adordd will do the rest.

VERY IMPORTANT

ALL ARRAYS ELEMENTS MUST BE IN UPPERCASE OTHERWISE INDEX WILL NOT WORK

B) SET ADO INDEX UDFS TO…

Here you must place the UDFs index expressions.

Indexes with UDFs are expensive to create and maintain so you should only place here your UDFs and index expressions that change the field length or value. Data type conversion expressions (the most used) should not be placed here.

Example:

SET ADO INDEX UDFS TO {“&”,”SUBSTR”,”PAD”,”MYFUNC”}

C) SET ADO TEMPORAY NAMES INDEX LIST TO…

Indicates the names used for temporary files at SQL level.

It must start by TMP or TEMP but can be "TMPROGER"

These temporary files are mainly used for temporary indexes created in the SQL server as TEMPORARY and automatically destroyed after connection ends.

They are only visible to the user that created them.

Example:

```
SET ADO TEMPORAY NAMES INDEX LIST TO
{"TMP","TEMP"}
```

D) SET ADO FIELDRECNO TABLES LIST TO …

This Set lets you indicate a different autoinc field of the default per table to be used as recno().

Example:

```
SET ADO FIELDRECNO TABLES LIST TO
{{"TABLE1","MYHBRECNO1"},{"TABLE2","MYHBRECNO2
"}}
```

E) SET ADO DEFAULT RECNO FIELD TO…

This Set indicates the default field name to be used as recno in all tables besides the mentioned above.

Example:

```
SET ADO DEFAULT RECNO FIELD TO "HBRECNO"
```

ATTENTION:

The D and or E sets are absolutely necessary and without them you will get an error trying to opening the table.

F) SET ADO FIELDDELETED TABLES LIST TO…

This Set lets you indicate a different logical field of the default per table to be used as deleted flag.

Example:

```
SET ADO FIELDDELETED TABLES LIST TO
{{"TABLE1","HBMYDELETE1"},{"TABLE2","HBMYDELET
E2"}}
```

G) SET ADO DEFAULT DELETED FIELD TO…

This Set indicates the default field name to be used as deleted flag in all tables besides the mentioned above.

Example:

```
SET ADO DEFAULT DELETED FIELD TO "HBDELETED"
```

ATTENTION:

The F and or G sets are absolutely necessary and without them you will get an error trying to opening the table.

H) SET ADO TABLES LOGICAL FIELDS LIST TO …

This array set is { { Tablename1, {boolean field1, booleanfield2,...} },;

{ tablename2, ...} }

To identify boolean fields for engines that don't have boolean type such as Oracle, Firebird, Sqlite and others.

In these cases there is no way to find out if a field is boolean therefore we need this set to let adordd know the boolean fields present in each table.

In order to avoid extra work for you when uploading tables **from only dbfcdx family of rdds** adordd builds itself the internal array, because through the dbfcdx rdd it knows if a field is boolean, in order that the uploading process doesn't errors.

At the same time adordd writes all this information in a file ext.ado to let you have the relation of all tables and boolean fields that you must place in this set next time app runs otherwise the results are unpredictable.

When creating new tables, not temporary, you must inform adordd through this set of the boolean fields.

With temporary tables it's not needed because at the time of the creation adordd keeps a record of boolean type fields but only during run time of the app.

If you are working with engines that have boolean type field you might forget this set.

I) SET ADO TABLES DECIMAL FIELDS LIST TO …

This array set is { { Tablename1, {cfield1, nDecimals,...} },;

{ tablename2, ...} }

Some engines like Access and SQlLite do not have specifically fields with decimal notation.

In these cases it was defaulting to 2 decimal places where could be more.

If the app would picture the gets with the fielddec function the nr of decimals places that could be entered was 2.

With this set we let adordd know the nr of decimals places per table and field.

When uploading tables that is done auto by adordd like for logical fields.

If you work with an engine that supports notation of decimals per field you might forget this set.

J) SET ADODBF INDEX LIST FIELDTYPE NUMBER TO …

This array set is {{ "Table",{ "numfiled", nlen },{"numfield", nLen } } }

adordd need a precise indication of the len of numeric fields used in index expressions.

This is only needed for numerica fields where the SQL type its without field len definition such as:

AUTOINC, MONEY, DOUBLE, INT, SMALLINT, etc.

If you don't use such fields as type "N" in index expression you can forget this set.

K) SET ADO DEFAULT DATABASE TO … SERVER TO … ENGINE TO … USER TO … PASSWORD TO … CLASSNAME …

This Set indicates the default server and database and authentication parameters we are using.

Connection gets established here.

The engines supported by ADORDD are ACCESS, MYSQL, ORACLE, INFORMIX, MSSQL, FIREBIRD, POSTGRE, ANYWHERE, DBASE, SQLITE, FOXPRO , ADS

Example:

```
// Access
SET ADO DEFAULT DATABASE TO
"D:\LUCAS\TEST2.mdb"
SERVER TO "" ENGINE TO ACCESS USER TO ""
PASSWORD TO  ""
```

```
// MySQL
SET ADO DEFAULT DATABASE TO cDataBase SERVER
TO cServer ENGINE TO MYSQL USER TO cUser
PASSWORD TO cPassWord
```

The CLASSNAME it's for future use. It allows having another class providing all ADO methods and Data to access through it the SQL engine.

L) SET ADO LOCK CONTROL ON / OFF

Enable disable lock control like DBFCDX in adordd.

M) SET ADO LOCK CONTROL SHAREPATH TO … RDD TO …

This set enables ADORDD to assure locking records and exclusive use of files as any other RDD.

If you want to leave the SQL engine take care of all locking you can use OFF but be aware that this does not conform with ISAM programing:

```
SET ADO FORCE LOCK OFF
```

You need to supply a path where ADORDD creates the tlocks.dbf file to control this.

This RDD file must be a RDD working with locks such as DBFCDX.

This is not a SQL table and if you need to work in WAN and need lock control you will need to:

a) The connection to SQL server

b) Ex a VPN where you can access this share.

Example:

```
SET ADO LOCK CONTROL SHAREPATH TO "D:\LUCAS"
RDD TO "DBFCDX"
```

N) SET ADO TABLENAME WITH PATH ON / OFF

This set enables ADORDD to treat table names with or without complete path.

There are cases were the database has complex file tree structure and same tables appear in different folders.

In these cases working with this set ON it will translate table name with path included to “path_tablename”.

Of course in these cases the uploading of the tables must be done in the same way. See HB_AdoRddUpload()

O) SET ADO ROOT PATH TO …. INSTEAD OF …..

This will allow you to upload the tables with certain path structure and then run the app with a different path.

If the tables names are with path on without this set the app wouldn’t run.

Example:

```
SET ADO ROOT PATH TO "D:\MYPATH" INSTEAD OF
"D:\ADORDD-TESTS"
```

Suppose that you uploaded all tables wit SET PATH ON and the path was “D: \ADORDD-TESTS” and now you run the app in the true app path which is “D:\MYPATH”.

Because the tables names have the path if you change the app path without this set app wouldn’t run anymore.

P) SET ADO CACHESIZE TO …. ASYNC … ASYNCNOWAIT ….

This set changes the way adordd gets the rows when opening the tables. Try different options and you will see the changes in performance mainly on huge tables.

Example:

```
SET ADO CACHESIZE TO 300 ASYNC ON ASYNCNOWAIT ON
```

This enables adordd to load the first 300 rows and the remaining asynchronously and the main ADO thread doesn't wait for the finish.

Be aware that all the rows might not be immediately available. You must try each option with your own application to check what is the best option for you. If you don't set these parameters adordd will use the default that assures total compatibility with your app although slower.

Q) SET ADO PRE OPEN THRESHOLD TO …. MASK ….

This set changes the way adordd open the tables.

All the tables with records equal or greater than nRecords and with values present in aMask in table name and without any WHERE opening clause will be all open and cached during app initialization.

You will wait a little for the app to start but after all tables opening will be very fast.

This will consume much memory at start but less during app runtime.

You should balance yourself the value of nRecords accordingly to the time you are willing to wait to open a table and your memory availability.

If you have several gigas of available memory you might choose to PRE OPEN all tables above 1000 records but if that's not the case may be you will have to increase this value only to PRE OPEN fewer tables.

Its very easy opening a couple of tables > 500.000 records to use some gigas of memory.

Try different values and you will see big changes in performance.

Example:

```
SET ADO PRE OPEN THRESHOLD TO 1000
```

This enables adordd to PRE OPEN all tables with more than 1000 records.

ATTENTION

Even without this set adordd will only take some time opening tables the first time. All successive openings will be very fast. This only applies when you open a table without WHERE clause.

Step 3: You can start working with ADORDD like a DBFCDX

and thats it!

REMARKS

For correct positioning the vertical bars in browses you need to call always OrdKeyNo and OrdKeyCount even when there isn't any active index.

This is because record recno() its not guaranteed to be sequential ordered as in DBF files.

Now if you need to upload your tables to any SQL you can do it with:

```
hb_AdoUpload( cBaseDir, cRDD, dbEngine, lOverWrite
)
```

Or you can do it also table by table:

```
    use "table" via "dbfcdx"
```

```
copy to "sqtable" via "adordd”

use "sqltable"

//you can use a table in a new connection

Use "ctable@connection string" alias
"whatever"
```

This is a low speed method and can take long time if the tables are huge. In this case you have to find yourself another way to upload data. You can use for example adofuncs.prg to import dbfs.

If you need to add deleted and/or recno field to your existing SQL tables you can do it like this:

```
oCon := hb_GetAdoConnection()

aTables :=  hb_adoRddGetTables( oCon )

FOR n := 1 to LEN( aTables )

 TRY // if already exist ignore it and continue

 oCon:Execute("ALTER TABLE "+( aTables[ n ] )+"
              ADD COLUMN HBDELETE BIT DEFAULT 0")

oCon:Execute("ALTER TABLE "+( aTables[ n ] )+"
              ADD COLUMN HBRECNO AUTOINC PRIMARY
KEY")

 CATCH

 END

NEXT
```

ADORDD FUNCTIONS:

ADOVERSION()
Returns adordd version

ADOBEGINTRANS(nWa)

ADOCOMMITTRANS(nWa)

ADOROLLBACKTRANS(nWa)

hb_AdoRddGetConnection(nWorkArea)

Returns the connection for the workarea

hb_AdoRddGetRecordSet(nWorkArea)

Returns the recordset for the nWorkArea

hb_AdoRddGetTableName(nWorkArea)

Returns table name for the nWorkArea

hb_AdoRddExistsTable(oCon,cTable, cIndex, cView)

Returns .t. if exist any on the DB

hb_AdoRddDrop(oCon, cTable, cIndex,cView, DBEngine)

Drops (delete) table, index or view in the DB

hb_GetAdoConnection()

Returns ado default connection

hb_AdoRddGetTables(oCon)

Returns all tables in the database in an array

hb_AdoUpload(cBaseDir, cRDD, dbEngine, lOverWrite)

Upload all tables in cBaseDir folder and all sub folders. If SET ADO TABLENAME WITH PATH ON the tables names will be translated to cPath_TableName otherwise only Tablename

HB_AdoRddFile(cFile)

Replacement of FILE() it checks the database for tables indexes and views and file system for other files.

hb_AdoRddDir(cPath)

Imitates the directory() but on a SQL database where table names are with path_tablename.

It returns an array with all table names.

hb_AdoRddCopyFile(cTableOrigin, cTableDestination)

Copy one table to another within the database. This can replace copyfile() one table to another. If working with PATH ON both parameters must include it.

ADORDD ERROR CODES:

10000 - "ADO needs field autoinc used as Recno " or "ADO needs field DELETED as logical used as DELETED()"

10002 - "record was deleted by other app and ADORDD cant reposition "

10200 - "Lock control ADO needs a share path and RDD to have it active "

10210 - "Lock control share path does not exist"

10500 - ""Connection to server/database not available""

Revision English manual: 18; Antonio H Ferreira; 07.12.2015